General Optimization Framework for Recurrent Reachability Objectives

David Klaška, Antónín Kučera*, Vít Musil and Vojtěch Řehák
Faculty of Informatics, Masaryk University, Brno, Czech Republic
david.klaska@mail.muni.cz, {tony, musil, rehak}@fi.muni.cz

Abstract

We consider the mobile robot path planning problem for a class of recurrent reachability objectives. These objectives are parameterized by the expected time needed to visit one position from another, the expected square of this time, and also the frequency of moves between two neighboring locations. We design an efficient strategy synthesis algorithm for recurrent reachability objectives and demonstrate its functionality on non-trivial instances.

1 Introduction

In mobile robot path planning, the terrain is represented as a directed graph where the vertices are robot positions, the edges correspond to possible robot moves, and every edge is assigned the corresponding traversal time. A moving strategy specifies how the robot moves from vertex to vertex, and it can be deterministic or randomized.

In this work, we concentrate on infinite-horizon path planning problems where the robot performs a recurring task such as surveillance or periodic maintenance. The standard tool for specifying infinite-horizon objectives are frequency-based objective functions parameterized by the limit frequency of visits to the vertices. Unfortunately, these functions are insufficient for expressing subtle optimization criteria used in specific areas such as robotic patrolling, and cannot faithfully capture all crucial properties of randomized strategies such as deviations/variances of relevant random variables. The latter deficiency represents a major problem often resolved indirectly by considering only deterministic strategies, even in scenarios where randomization achieves strictly better performance and is easy to implement (see Example 1).

Our Contribution. We design and investigate a class of recurrent reachability objective functions based on the following parameters:

1. the limit frequency of edges;
2. the expected time to hit a given set of vertices from another given vertex;
3. the expected square of the time to hit a given set of vertices from another given vertex.

Note that using (1), one can express the frequency of visits to vertices, and (2) and (3) allow to express the variance and standard deviation of the time to hit a given set of vertices from another given vertex. Thus, the recurrent reachability objective functions can “punish” large deviations from the expected values, allowing for balancing performance with stochastic stability.

Computing an optimal moving strategy for a given recurrent reachability objective is computationally hard. One can easily reduce the NP-hard Hamiltonian cycle problem to the problem of deciding whether the minimum of a certain recurrent reachability objective function is bounded by a given constant. This means there is no efficient strategy synthesis algorithm with optimality guarantees unless \( P = NP \).

We design a strategy synthesis algorithm based on gradient descent applicable to arbitrary recurrent reachability objectives involving piecewise differentiable continuous functions. The algorithm efficiently computes a finite-memory randomized strategy where the memory is used to “remember” some relevant information about the history of visited vertices. Although the (sub)optimality of this strategy is not guaranteed for the reasons mentioned above, our experiments show that the algorithm can solve instances requiring non-trivial insights and produce solutions close to theoretical optima.

Thus, we obtain a general and efficient optimization framework for an expressively rich class of non-linear infinite-horizon objectives capable of solving problems beyond the reach of existing methods.

1.1 Motivating Example

In this section, we give an example illustrating the limitations of frequency-based objectives and deterministic strategies, and we show how randomization and recurrent reachability objectives help to overcome these problems.

In robotic patrolling, some vertices in the terrain graph are declared as targets, and the robot aims to discover possible intrusions at the targets. One standard measure for the protection achieved by a given moving strategy is the maximal average idleness of a target [Huang et al., 2019; Almeida et al., 2004; Portugal and Rocha, 2011]. In the language of Markov chains, this corresponds to the maximal renewal time of a target, i.e., \( \max_{\tau \in T} 1/f_\tau \), where \( T \) is the set of all targets and \( f_\tau \) is the frequency of visits to \( \tau \) (recall that \( 1/f_\tau \) is the expected time of revisiting \( \tau \)).
1.2 Related Work

The finite-horizon path planning problem involving finding a feasible path between two given positions is one of the most researched subject in mobile robotics (see, e.g., [Choset, 2005; LaValle, 2006]). Recent technological advances motivate the study of infinite-horizon path planning problems where the robot performs an uninterrupted task such as persistent data-gathering [Smith et al., 2011] or patrolling [Huang et al., 2019; Almeida et al., 2004; Portugal and Rocha, 2011]. The (classical) vehicle routing problem and the generalized traveling salesman problem [Toth and Vigo, 2001] can also be seen as infinite-horizon path planning problems. The constructed strategies were originally meant for humans (drivers, police squads, etc.) and hence they are deterministic. The only exception is adversarial patrolling based on Stackelberg equilibria [Sinha et al., 2018; Yin et al., 2010] where randomization was soon identified as a crucial tool for decreasing the patroller’s predictability.

The existing objectives studied in infinite-horizon path planning problems are mostly based on long-run average quantities related to vertex frequencies or cycle lengths, such as mean payoff or renewal time (see, e.g., [Puterman, 1994]). General-purpose specification languages for infinite-horizon objectives are mostly based on linear temporal logics (see, e.g., [Patrizi et al., 2011; Wolff et al., 2014; Ulusoy et al., 2014; Bhatia et al., 2010]). A formula of such a logic specifies desirable properties of the constructed path, and may include additional mechanisms for identifying a subset of optimal paths. The optimality criteria are typically based on distances between vertices along a path, and the constructed strategies are deterministic.

To the best of our knowledge, the recurrent reachability objective functions represent the first general-purpose specification language allowing to utilize the benefits of randomized strategies and even specify tradeoffs between performance and stochastic stability. Even in a broader context of stochastic programming, the existing works about balancing quantitative features of probabilistic strategies have so far been limited to some variants of mean payoff and limit frequencies [Brázdil et al., 2017]. These results are not applicable in our setting due to a different technical setup. Furthermore, our specification language is not limited to frequency-based objectives.

2 The Model

In the rest of this paper, we use \( \mathbb{N} \) and \( \mathbb{N}_+ \) to denote the sets of non-negative and positive integers. The set of all probability distributions over a finite set \( A \) is denoted by \( \text{Dist}(A) \). A distribution \( \nu \) over \( A \) is positive if \( \nu(a) > 0 \) for all \( a \in A \), and Dirac if \( \nu(a) = 1 \) for some \( a \in A \).

A finite-state Markov chain is a pair \( M = (S, P) \) where \( S \) is a finite set of states and \( P : S \times S \rightarrow [0, 1] \) is a stochastic matrix where the sum of every row is equal to 1. A bottom strongly connected component (BSCC) of \( M \) is a maximal \( B \subseteq S \) such that for all \( s, t \in B \) we have that \( s \) can reach \( t \) with positive probability (i.e., \( P^m(s, t) > 0 \) for some \( m \in \mathbb{N} \)), and every \( t \) reachable from a state of \( B \) belongs to \( B \).

We assume familiarity with basic results of ergodic theory and calculus that are recalled at appropriate places.

2.1 Terrain Model

The terrain is modeled as a finite directed graph \( G = (V, E, \text{tm}) \) where the vertices of \( V \) correspond to robot’s positions, the edges \( E \subseteq V \times V \) are possible moves, and \( \text{tm} : E \rightarrow \mathbb{N}_+ \) specifies the traversal time of an edge. Later,
we adjoin additional parameters to edges and vertices modeling the costs, importance, etc.

We require that $G$ is strongly connected, i.e., for all $v, u \in V$ there is a path from $v$ to $u$. We write $u \rightarrow v$ instead of $(u, v) \in E$.

2.2 Moving Strategy

Let us fix a graph $G = (V, E, \tau m)$, and let $M$ be a non-empty set of memory states. Intuitively, memory states are used to "remember" some information about the history of visited vertices. Since infinite-memory is not implementable, from now on we restrict ourselves to finite-memory strategies.

A robot’s configuration $(v, m)$ is determined by the currently visited vertex $v$ and the current memory state $m$. We use Conf $= V \times M$ to denote the set of all configurations. A configuration of the form $(v, m)$ is written as $\hat{v}$.

When the robot visits a vertex, the next move is chosen randomly according to the current configuration. Formally, a moving strategy for $G$ with memory $M$ is a function $\sigma$: Conf $\rightarrow \text{Dist}(\text{Conf})$ such that $\sigma(\hat{v}) > 0$ only if $v \rightarrow u$. If $\sigma(\hat{v})$ is a Dirac distribution for every $\hat{v}$ in Conf, we say that $\sigma$ is deterministic.

Every moving strategy $\sigma$ determines a Markov chain $G_\sigma$ where Conf is the set of states and $P(\hat{v}, \hat{u}) = \sigma(\hat{v})(\hat{u})$. The edges of $G_\sigma$ are defined by $\hat{v} \rightarrow \hat{u}$ iff $v \rightarrow u$. Note that some edges may have zero probability, i.e., $\sigma(\hat{v})(\hat{u}) = 0$, but the BSCCs of $G_\sigma$ are determined by the edges with positive probability (see the above definition of Markov chain). The traversal time of an edge $\hat{v} \rightarrow \hat{u}$ is the same as in $G$, i.e., equal to $\tau m(v, u)$.

2.3 Recurrent Reachability Objectives

For the rest of this section, we fix a graph $G = (V, E, \tau m)$ and a finite set $M$ of memory states.

Atomic Expressions

We start by introducing basic expressions used to construct recurrent reachability objectives.

A walk in $G_\sigma$ is an infinite sequence $w = \hat{v}_0, \hat{v}_1, \ldots$ such that $\hat{v}_i \rightarrow \hat{v}_{i+1}$ for all $i \in \mathbb{N}$. Let $C \subseteq \text{Conf}$. We say that $w$ hits $C$ in time $t$ if $t = \sum_{i=0}^{m-1} \tau m(v_i, v_{i+1})$, where $m$ is the least index $j$ such that $\hat{v}_j \in C$. If there is no such $j$, we put $t = \infty$. For all $\hat{v} \in \text{Conf}$ and $C \subseteq \text{Conf}$, we use $T(\hat{v}\rightarrow C)$ to denote the expected hitting time of $C$ by a walk initiated in $\hat{v}$ (if $\hat{v} \in C$, then $T(\hat{v}\rightarrow C) = 0$), and $T^2(\hat{v}\rightarrow C)$ to denote the expected square of the hitting time of $C$ by a walk initiated in $\hat{v}$. Furthermore, for every walk $w = \hat{v}_0, \hat{v}_1, \ldots$ and every edge $\hat{e} = \hat{v} \rightarrow \hat{u}$, we define the frequency of $\hat{e}$ along $w$ as the limit percentage of time spent by executing $\hat{e}$ along $w$, i.e.,

$$F(\hat{e}, w) = \lim_{m \rightarrow \infty} \frac{\tau m(v, u) \cdot \#_w(\hat{v}_0, \ldots, \hat{v}_{m+1})}{\sum_{i=0}^{m} \tau m(v_i, v_{i+1})},$$

(1)

where $\#_w(\hat{v}_0, \ldots, \hat{v}_{m+1})$ is the number of occurrences of $\hat{e}$ in the prefix $\hat{v}_0, \ldots, \hat{v}_{m+1}$. It follows from basic results of Markov chain theory that the above limit is defined for almost all walks (i.e., with probability one), and almost all walks $w$ that hit the same BSCC of $G_\sigma$ have the same $F(\hat{e}, w)$.

Syntax

Recurrent reachability objective functions are closed-form expressions over numerical constants and atomic expressions of the form $T(\hat{e} \rightarrow C)$, $T^2(\hat{e} \rightarrow C)$, $F(\hat{e})$, and $p(\hat{e})$ obtained by using

- addition, multiplication, min, and max that may take arbitrarily many arguments;
- division, where the denominator is an expression over numerical constants and atomic expressions of the form $F(\hat{e})$, $p(\hat{e})$ built using addition and multiplication.
- other differentiable functions such as square root that are defined for all non-negative arguments.

When defining the arguments of sums, products, min, and max, we may refer to special sets $A_e$ and $A_C$ consisting of active configurations and edges, respectively, whose semantics is defined in the next paragraph.

A recurrent reachability optimization problem is a problem of the form $\text{minimize } R$ or $\text{maximize } R$, where $R$ is a recurrent reachability objective function.

Ergodicity

Infinite-horizon objective functions are typically independent of finite prefixes of runs and the initial configuration can be chosen freely. Hence, the objective value for $G_\sigma$ actually depends only on the objective values attained in the “best” BSCC of $G_\sigma$. From now on, we only consider recurrent reachability objective functions $R$ satisfying this condition (this is equivalent to requiring that $R$ can be optimized by an ergodic strategy where $G_\sigma$ is strongly connected).

Evaluation

Let $\sigma$ be a moving strategy for $G$ with memory $M$, and let $B$ be a BSCC of $G_\sigma$. For a given recurrent reachability function $R$, we use $R[B]$ to denote the value of $R$ in $B$, defined by structural induction as follows:

- atomic expressions $T(\hat{e} \rightarrow C)$, $T^2(\hat{e} \rightarrow C)$, $F(\hat{e})$, and $p(\hat{e})$ are evaluated in the way described above ($p(\hat{e})$ is the probability of $\hat{e}$ assigned by $\sigma$). In particular, note that $F(\hat{e})$ where $\hat{e} = \hat{e} \rightarrow \hat{u}$ can be positive only if $\hat{v}, \hat{u} \in B$.
- The set $A_e$ of active configurations is equal to $B$, and the set $A_e$ of active edges consists of all $(\hat{v}, \hat{u})$ such that $\hat{v}, \hat{u} \in B$ and $\sigma(\hat{v})(\hat{u}) > 0$.
- The addition, multiplication, min and max are evaluated in the expected way. If the set of arguments is parametrized by $A_e$, or $A_C$, it is constructed for the set of configurations and edges defined in the previous item.

In some cases, $R[B]$ can be undefined (see Section 4).

Finally, we define the $\sigma$-value of the objective $\text{minimize } R$ or $\text{maximize } R$ as the minimal or the maximal $R[B]$ such that $B$ is a BSCC of $G_\sigma$ where $R[B]$ is defined (if there is no such $B$, then the $\sigma$-value in undefined).

3 Examples

To demonstrate the versatility of recurrent reachability objectives, we present selected examples of concrete objective functions. The list is by no means exhaustive—we show how
to capture some of the existing infinite-horizon optimization criteria and how to extend them to control various forms of stochastic instability caused by randomization. Let us emphasize that our aim is to illustrate the expressive power of our optimization framework, not to design the most appropriate objectives capturing the discussed phenomena.

For this section, we fix a graph $G = (V, E, tm)$ and a finite set $M$ of memory states. For a given subset $U \subseteq V$, we use $C_U$ to denote the subset $U \times M$ of configurations.

For simplicity, in the first two subsections we assume that the traversal time of every edge is 1.

**Mean Payoff**

As a warm-up, consider the concept of mean payoff, i.e., the long-run average payoff per visited vertex. Let $\alpha : V \rightarrow N_+$ be a payoff function. The goal is to minimize the mean payoff for $\alpha$. This is formalized as **minimize** $MP$, where

$$
MP \equiv \sum_{v \in \text{Conf}} F(\hat{v}) \cdot \alpha(v),
$$

(2)

$$
F(\hat{v}) \equiv \sum_{\hat{v} \in \text{Out}(\hat{v})} \mathbb{E}(\hat{v}).
$$

(3)

Here, $\text{Out}(\hat{v})$ is the set of all out-going edges of $\hat{v}$. Hence, $F(\hat{v})$ is the limit frequency of visits to $\hat{v}$.

Minimizing mean-payoff is computationally easy. However, the costs of vertices visited by an optimal strategy can significantly differ from the mean payoff (i.e., the costs are not distributed sufficiently “smoothly” along a walk). If “smoothness” is important, it can be enforced, e.g., by the objective **minimize** $MP + \beta \cdot DMP$, where $\beta$ is a suitable weight and

$$
DMP \equiv \sqrt{\sum_{v \in \text{Conf}} (F(\hat{v}) \cdot (MP - \alpha(v)))^2}
$$

(4)

is the standard deviation of the costs per visited vertex. Similarly, we can formalize objectives involving multiple cost/payoff functions and enforce some form of “combined smoothness” if appropriate.

**Renewal Time**

Let $T \subseteq V$ be a set of targets, and consider the problem of minimizing the maximal renewal time of a target. First, for every $\hat{\tau} \in C^T$, let

$$
P(\hat{\tau}) \equiv \frac{F(\hat{\tau})}{\sum_{\kappa \in C^T} F(\kappa)},
$$

(5)

where $F(\cdot)$ is defined by (3). Hence, $P(\hat{\tau})$ is the percentage of visits to $\hat{\tau}$ among all visits to configurations of $C^T$, assuming that the denominator is positive. Then, the renewal time of $\tau$, denoted by $\text{ERen}(\tau)$ is given by

$$
\sum_{\hat{\tau} \in C^T} P(\hat{\tau}) \sum_{\hat{\tau} \rightarrow \hat{u}} \pi(\hat{\tau}, u) \cdot (1 + \mathbb{T}(\hat{u} \rightarrow C^T))
$$

(6)

and the expected square of the time needed to revisit $\tau$, denoted by $\text{QRen}(\tau)$, is expressible as

$$
\sum_{\hat{\tau} \in C^T} P(\hat{\tau}) \sum_{\hat{\tau} \rightarrow \hat{u}} \pi(\hat{\tau}, u) \cdot \left(1 + 2\mathbb{T}(\hat{u} \rightarrow C^T) + \mathbb{T}^2(\hat{u} \rightarrow C^T)\right).
$$

(7)

Minimizing the maximal renewal time is then formalized as **minimize** $\max_{\tau \in T} \text{ERen}(\tau)$. However, this simple objective does not take into account possible deviations of the renewal time from its mean. This can be captured, e.g., by expressing the standard deviation as

$$
\text{DevRen}(\tau) \equiv \sqrt{\text{QRen}(\tau) - (\text{ERen}(\tau))^2}
$$

(8)

and using **minimize** $\max_{\tau \in T} (\text{ERen}(\tau) + \beta \cdot \text{DevRen}(\tau))$ for a suitable weight $\beta$.

**Patrolling**

Let $T \subseteq V$ be a set of targets, and $\alpha : T \rightarrow N_+$ a function assigning to every target its importance. The damage caused by an attack at a target $\tau$ (such as setting a fire) is given as $\alpha(\tau) \cdot t$ where $t$ is the time to discover the attack by the robot.

The patrolling objective is to minimize the expected damage.

Patrolling problems are studied for adversarial and non-adversarial environments. In the first case, there is an active Attacker knowing the robot’s strategy and observing its moves. For the Attacker, an appropriate moment to initiate an attack is when the robot leaves a vertex and starts walking along some edge $\hat{v} \rightarrow \hat{u}$. The objective is to minimize the expected damage over all possible attacks, i.e.,

$$
\text{minimize} \max_{\hat{v} \rightarrow \hat{u} \in \text{At}.} \max_{\tau \in T} \alpha(\tau) \cdot (\mathbb{E}(\text{tm}(v, u) + \mathbb{T}(\hat{u} \rightarrow C^\tau))).
$$

(9)

Note that (9) conveniently uses the set $\text{At}$. of active edges to restrict the $\max$ only to “relevant” edges used by the strategy.

In non-adversarial patrolling, the attacks are performed by “nature”. Let $\pi$ be a distribution over $T$ specifying the attack chance (such as the probability of spontaneous ignition). Then, minimizing the expected damage is expressible as **minimize** $\text{EDam}$, where the function $\text{EDam}$ is defined as

$$
\sum_{\hat{v} \rightarrow \hat{u} \in \text{At}.} \mathbb{E}(\hat{v}, \hat{u}) \sum_{\tau \in T} \pi(\tau) \alpha(\tau) \left(\frac{\mathbb{E}(\text{tm}(v, u))}{2} + \mathbb{T}(\hat{u} \rightarrow C^\tau)\right).
$$

(10)

Note that if $\tau$ is attacked when the robot walks along $\hat{v} \rightarrow \hat{u}$, then the robot is in the middle of this edge on average. Hence, the average time to reach $\tau$ is $\mathbb{E}(\text{tm}(v, u))/2 + \mathbb{T}(\hat{u} \rightarrow C^\tau)$.

Again, we can express the variances/deviations of the relevant random variables (incl. the variance of the expected damage of (10)). These expressions are relatively long, but their construction is straightforward.

4 The Algorithm

The algorithm is based on gradient descent: It starts with a random initial strategy and then repeatedly evaluates the objective function $R$ and modifies the current strategy in the direction of the gradient $\nabla R$ (or $-\nabla R$ for minimization). After a number of iterations, the strategy attaining the best objective value is returned.

Let $\sigma$ be a moving strategy for a graph $G$. We show how to evaluate and differentiate $R[B]$ for a given BSCC $B$ of $G_s$. The atomic expressions are obtained as unique solutions of linear equations systems. More concretely, for a target set $C$, active configurations $\text{At}$. and edges $\text{At}.$, we have that $\mathbb{T}(\hat{v} \rightarrow C)$ and $\mathbb{T}^2(\hat{v} \rightarrow C)$ are undefined for $\hat{v} \notin \text{At}$. If $\hat{v} \in \text{At}$. and $\text{At} \cap C = \emptyset$, then $\mathbb{T}(\hat{v} \rightarrow C) = \mathbb{T}^2(\hat{v} \rightarrow C) = \infty$.
If $\mathcal{A}_c \cap C \neq \emptyset$, then for every $\hat{v} \in \mathcal{A}_c$ we fix a variable $X_{\hat{v}}$ and an equation

$$X_{\hat{v}} = \begin{cases} 0 & \text{if } \hat{v} \in C, \\ \sum_{\hat{v} \to \hat{w}} \sigma(\hat{v})(\hat{w}) \cdot (tm(v, w) + X_{\hat{w}}) & \text{otherwise}. \end{cases}$$

Then, the tuple of all $\mathcal{T}(\cdot \to C)$, where $\hat{v} \in \mathcal{A}_c$, is the unique solution of this system.

Similarly, if $\mathcal{A}_c \cap C \neq \emptyset$, then the tuple of all $\mathcal{T}(\cdot \to C)$, where $\hat{v} \in \mathcal{A}_c$, is the unique solution of the system where to each $\hat{v} \in \mathcal{A}_c$, we assign a variable $Y_{\hat{v}}$ and an equation

$$Y_{\hat{v}} = \begin{cases} 0 & \text{if } \hat{v} \in C, \\ \sum_{\hat{v} \to \hat{w}} \sigma(\hat{v})(\hat{w}) \cdot (\gamma(\hat{v}, \hat{w}) + \gamma(\hat{v}, \hat{w})) & \text{otherwise}, \end{cases}$$

where $\gamma(\hat{v}, \hat{w}) = tm(v, w)(tm(v, w) + 2\mathcal{T}(\hat{w} \to C)) + Y_{\hat{w}}$.

To compute the edge frequencies, we fix a variable $Z_{\hat{v}}$ for every $\hat{v} \in \mathcal{A}_c$, and an equation

$$Z_{\hat{v}} = \sum_{\hat{v} \to \hat{w}} \sigma(\hat{v})(\hat{w}) \cdot Z_{\hat{w}}. \quad \text{This system, together with the equation } \sum_{v \in \mathcal{A}_c} Z_{\hat{v}} = 1, \text{ has a unique solution } F \text{ where } F(\cdot \to C) \text{ is the frequency of visits to } \hat{v}. \quad \text{For each edge } \hat{e} = (\hat{v}, \hat{w}) \in \mathcal{A}_c, \text{ we set } D(\hat{e}) = F(\hat{v}) \cdot \sigma(\hat{v})(\hat{w}) \cdot tm(v, w) \text{ and we get}$$

$$F(\cdot) = D(\cdot) / \sum_{\hat{e} \in \mathcal{A}_c} D(\hat{e}).$$

For the other edges $\hat{e} \notin \mathcal{A}_c$, we have that $F(\cdot \to C) = 0$. The value of $R[\hat{B}]$ is then obtained from the atomic expressions in the straightforward way (when some atomic expression used in $R$ is undefined or the denominator of some fraction of $R$ is zero in $B$, then $R[\hat{B}]$ is undefined).

Next, we need to calculate the gradient $\nabla R$. As objectives are, by design, allowed to use only smooth operations with min and max over atomic expressions, the derivatives of $R$ w.r.t. these atomic expressions are well defined almost everywhere. Solutions of systems of linear equations depend smoothly on the parameters and the derivatives of our atomic expressions w.r.t. $\sigma$ can be calculated as solutions of other linear systems. We use PyTorch Library [Paszke et al., 2019] and its automatic differentiation in our implementation.

However, a naïve update $\sigma \pm \lambda \nabla R$ for a step size $\lambda$ almost never yields a probability distribution (i.e., a valid strategy). The standard approach is to produce strategies from real-valued coefficients by a Softmax function. Any update of these coefficients then leads to a well-defined strategy. The drawback of Softmax is that the resulting distributions never contain zeros (i.e., the strategies always use all edges).

To reach all possible strategies, we cut the small probabilities at a certain threshold (and normalize) by Cutoff function. However, as edges with zero probabilities are excluded from $\mathcal{A}_c$, discontinuities in the objective may occur. For instance, in Patrolling objective (9), the term $tm(u, u)$ is present for an edge $\hat{v} \to \hat{u}$ if $\sigma(\hat{v})(\hat{u}) > 0$ and drops to zero if $\sigma(\hat{v})(\hat{u}) = 0$.

In other words, objective $R$ as a function of real-valued coefficients is a smooth function on an open set with possible jumps at the boundary. In order to make the boundary values accessible by the gradient descent, we relax our discontinuous $R$ to a smooth one, say $R^*$. For instance, in Patrolling objective (9), we multiply $tm(u, u)$ by a factor $\text{HardTanh}(\epsilon \sigma(\hat{v}, \hat{u}))$, which interpolates the values 0 and 1 continuously. Moreover, for a more efficient gradient propagation, we replace each $\min$ and $\max$ with their relaxed variants as in [Klaska et al., 2018].

The final algorithm is described in Algorithms 1. Strategy coefficients are initialized at random. In every step, we compute the relaxed objective $R^*$ value of the current strategy and update the coefficients based on the gradient $\nabla R^*$ using Adam optimizer [Kingma and Ba, 2015]. We also add a decaying Gaussian noise to the gradient. Then, we round the strategy using Cutoff and compute its true objective value $R$. Algorithms 1 can be run repeatedly to increase the chance of producing a strategy with better value.

The algorithm is efficient because it does not involve any computationally demanding tasks, but it does not guarantee (sub)optimality of the constructed strategies. This is unavoidable due to the NP-hardness of some of the recurrent reachability objectives.

5 Experiments

There is no previous work setting the baseline for evaluating the quality of strategies produced by our algorithm. Therefore, we selected two representative examples where the tradeoffs between performance and stability are not easy to discover, but the functionality of the synthesized strategies can still be analyzed by hand and compared against the best achievable outcomes identified by theoretical analysis.1

Mean Payoff

We minimize the objective $MP + \beta \cdot DMP$ defined in (2) and (4) for the graph of Fig. 2. The graph contains three cycles with the corresponding MP and DMP as in the table of Fig. 2. For almost every $\beta$, the objective’s minimizer is precisely one of the three cycles. More precisely, it is the cycle 7-0-5-7 for all $\beta \in [0, \varrho_1]$, the cycle 7-6-5-7 for all $\beta \in [\varrho_1, \varrho_2]$, and the cycle 7-6-7 for all $\beta \in [\varrho_2, \infty)$, where $\varrho_1 \approx 0.73$ and $\varrho_2 \approx 1.16$. Ideally, our algorithm should find the corresponding cycle for every $\beta$.

1The code for reproducing the results is available at https://gitlab.fi.muni.cz/formela/2022-ijcai-optimization-framework. See also the latest version at https://gitlab.fi.muni.cz/formela/regstar.
consider the graph of Fig. 4a. We minimize the objective

$$\text{max}_{\tau \in T} (\text{ERen}(\tau) + \beta \cdot \text{DevRen}(\tau))$$

defined in (6) and (8).

The outcomes of our algorithm for two memory states are

shown in Fig. 5. For each \( \beta \) ranging from 0 to 0.3, we run

100 trials and report the expected renewal time and the cor-

responding standard deviation of the resulting 100 strategies; the

best values are highlighted by solid dots. The values of the

obtained strategies are concentrated near the best one, show-

ing the stability of the optimization.

For smaller \( \beta \), the constructed strategies have a smaller

renewal time but large deviation, and they work as shown in

Fig. 4b. That is, they tend to re-visit \( \tau_1 \) from \( v_1 \) and \( \tau_2 \) from

\( v_2 \). As \( x, y \to 1 \), the maximal renewal time approaches 40

and the standard deviation approaches \( \infty \).

For larger \( \beta \), where the standard deviation is punished more

severely, the algorithm tends to decrease \( x, y \). Note that for

\( x = y = 0 \), the strategy of Fig. 4b becomes deterministic

with renewal time 60 and zero deviation. However, this point

is not reached by the curve of Fig. 5. The reason is that for

\( \beta \approx 0.27 \), the algorithm discovers and strongly prefers a

completely different strategy, which goes through \( v_3 \) instead (see

Fig. 4c), with maximal renewal time 52 and zero deviation.

### 6 Conclusion

The obtained optimization framework for recurrent reacha-

bility objectives is applicable not only to graphs, but also to

Markov decision processes without any additional effort. Here,

the randomization introduced by the strategies is com-

bined with the “inherent randomization” of the model. Here,

stochastic instability cannot be removed completely, and a

precise understanding of the principal limits of this effort is a

challenging direction for future research.

### Acknowledgements

This work is supported by the Czech Science Foundation,

Grant No. 21-24711S, and from Operational Programme Re-

search, Development and Education – Project Post-

doc2MUNI No. CZ.02.2.69/0.0/0.0/18_053/0016952.
References


